Patrones en la Ingeniería del Software

Por Ing. Jorge A. Rodríguez Zamora
Jefe de Grupo de Gestión y Admin. de Aplicaciones
Gerencia Territorial Camagüey, ETECSA
jrz@cmg.tel.etecsa.cu

La Ingeniería del Software

El siglo XXI se enfrenta inexorablemente al acelerado desarrollo de la sociedad del conocimiento. Esta nueva era no sólo está cambiando la sociedad, sino que los nuevos modelos de negocios requieren la reformulación de conceptos. Conocimiento, información, Internet, Web son algunos de los términos más utilizados en cualquier ambiente o negociación, por lo tanto, son necesarias nuevas tendencias apoyadas precisamente en el conocimiento. La Ingeniería de Software no es una excepción, y requiere no sólo una actualización de conceptos, sino también la comprensión y formulación del conocimiento existente entorno a las nuevas tecnologías e innovaciones teóricas que se encuentren en desarrollo dinámico .

Constituye una importante disciplina dentro del área de la Informática y las Ciencias de la Computación, encargada de ofrecer métodos y técnicas para la creación y el mantenimiento de aplicaciones de software con elevados niveles de calidad. Se emplea para el desarrollo de diferentes productos informáticos, aborda todas las fases del ciclo de vida del proyecto de desarrollo y es flexiblemente aplicable a entornos de investigación, producción, aplicaciones bancarias, ciencias médicas, redes de datos, entre otros [18].

Sin técnicas de ingeniería es prácticamente imposible sustentar el consenso general que plantea la necesidad de producir aplicaciones informáticas basadas en modelos estándares. “La Ingeniería del Software, constituye la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, funcionamiento y mantenimiento del software; es decir, la aplicación de ingeniería al software”[14].

La Ingeniería del Software es una tecnología multicapa, por lo tanto, cualquier enfoque de ingeniería, incluida la aplicada al software, debe sustentarse sobre un compromiso de organización y calidad [18]. Una vez basada su estructura sobre la organización y la calidad, el fundamento esencial es la capa de proceso que constituye la unión de las capas tecnológicas y permite el desarrollo racional y oportuno de la Ingeniería del Software. “El proceso define un marco de trabajo para un conjunto de áreas claves de procesos que se deben establecer para la entrega efectiva de la tecnología de la ingeniería del software”[17]. Las áreas claves del proceso forman la base del control de gestión de proyectos del software y establecen el contexto en el que se aplican los métodos técnicos, se obtienen productos del trabajo —modelos, documentos, datos, informes, formularios—, se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente [18].

Entre los procesos que se identifican en el desarrollo de un software se encuentran los relacionados con individuos aislados —Proceso Personal de Software (PSP)— y al trabajo en equipo —Proceso de Software en Equipos (TSP)—. Además, se conforma un modelo que establece el nivel de madurez del proceso en toda organización —Modelo de Madurez de las Capacidades (CMM)—, como un estándar de calidad.

Los métodos de la Ingeniería de Software indican cómo construir técnicamente un software, abarcan las tareas que incluyen análisis de requerimientos, diseño, programación, pruebas y mantenimiento. También dependen de los princi-

pios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelación y otras técnicas descriptivas [18].

Actualmente, existen diversos métodos de Ingeniería de Software que fundamentan su utilidad en la capacidad para facilitar las tareas de análisis, diseño e implementación. Para la modelación de sistemas, ha surgido como estándar el Lenguaje Unificado de Modelado (UML); y las metodologías que conducen a todo proyecto a través de las diferentes etapas que conforman el proceso de desarrollo del software, al otorgarles calidad y eficiencia, como ADOOSI-UML y el Proceso Unificado Racional (RUP). También hay técnicas establecidas para potenciar y facilitar la labor de los especialistas, por ejemplo, el empleo de Patrones / AntiPatrones.

Las herramientas de la Ingeniería del Software proporcionan un enfoque automático o semiautomático para el proceso y para los métodos. Cuando se integran herramientas para que la información creada por una de ellas pueda utilizarla otra, se establece un sistema de soporte para el desarrollo de software. Entre las más utilizadas, se encuentran las CASE —*Computer Aided Software Engineering* / Ingeniería del Software Asistida por Computadora—. Estas son tecnologías para automatizar el desarrollo y mantenimiento del software, combinan herramientas de software y metodologías, y deben constituir un conjunto integrado que automatice todas las partes del ciclo de vida y ahorren trabajo a los especialistas.

Además de las CASE, han surgido otros tipos de herramientas como los CAME —*Computer Aided Method Engineering* / Método de Ingeniería Asistido por Computadora—, IPSE —*Integrated Project Support Environment* / Entorno de Apoyo al Proyecto Integrado—, SEE —*Software Engineering Environment* / Entorno de Ingeniería de Software—, CSCW —*Computer Supported Cooperative Work* / Trabajo Colaborativo Apoyado en Computadora— y los WFMS —*Workflow Management Systems* / Sistemas de Administración del Flujo de Trabajo— [12].

Patrones en la Ingeniería del Software

La noción genérica de los patrones se origina a partir de los trabajos desarrollados en los años setenta por Christopher Alexander, prestigioso arquitecto en construcción civil, quien describe, en uno de sus más reconocidos textos, algo catalogado como "una cualidad sin nombre (…) es la cosa más preciada que existe en nuestras vidas" [1], refiriéndose a lo que posteriormente bautizó con el nombre de patrón.

En sus escritos, Alexander expone sus puntos de vista con relación a problemas recurrentes que descubre en la construcción de ciudades y pueblos, describe dichos problemas, sus respectivas soluciones, y hace uso de la expresión que él mismo definió "Cada patrón describe un problema que ocurre más de una vez en nuestro entorno, y además, describe la solución a dicho problema, de forma tal que pueda ser utilizada dicha solución siempre que se vuelva nuevamente a la misma situación"[2].

La aplicación de los patrones en el desarrollo de software es uno de los tópicos más debatidos en los últimos años dentro de la comunidad de investigadores, arquitectos y programadores de aplicaciones de software. "Es fundamental para cualquier ciencia o especialidad de ingeniería que exista un vocabulario común para la expresión de sus conceptos, así como un lenguaje que relacione a los mismos"[4]. La principal ambición del uso de los patrones dentro de la comunidad del software es la creación de un cuerpo literario capaz de ayudar a sus profesionales a dilucidar con claridad los problemas recurrentes encontrados durante el desarrollo de software.

Conceptualmente, la esencia de un patrón es la descripción de un problema y su solución. El problema es elaborado en términos de un contexto y de fuerzas aplicables en él. El rol de la solución es declarar las fuerzas para que se generen consecuencias positivas, beneficios y problemáticas manejables, y se conduzcan hacia la aplicación de otros patrones. La condición necesaria para ser considerado un patrón es la observación exitosa en, al menos, tres sistemas reales de la solución que ofrece. Como tres acontecimientos no pueden ser idénticos, los patrones son abstracciones de sus experiencias y resultados [6].

Linda Rising expresa en uno de sus libros: "(…) un patrón no es más que otra manera de documentación" [20]. El poder de este tipo de documentación es que el conocimiento, previamente adquirido por desarrolladores con experiencia, es capturado en una forma aceptable y puesto al alcance de todos. Los patrones son objetos que han sido descubiertos en más de un sistema y proporcionan soluciones exitosas a problemas recurrentes.

A pesar de que la mayoría de los estudios han estado dirigidos al desarrollo orientado a objetos, y publicaciones y descubrimientos de patrones han sido realizados sobre este paradigma, la noción de patrones no se encuentra vinculada a ninguna metodología o lenguaje. Muchos trabajos sobre patrones han sido desarrollados en sistemas no orientados a objetos. "Los patrones no son un

concepto orientado a objetos" [20].

Los patrones ayudan a la creación y consolidación de un lenguaje compartido que permita la comunicación de experiencias sobre los problemas y sus respectivas soluciones. Si se codifican formalmente estas soluciones y las relaciones entre ellas, es posible capturar exitosamente el conjunto de conocimientos, para definir las nociones de buena arquitectura y diseño. "Formando un lenguaje de patrones común para la comunicación de estructuras y mecanismos sobre nuestra arquitectura, nos permite razonar claramente en función de ello"[4].

Elementos para la creación de patrones

Para la creación de nuevos patrones debe considerarse, en primer lugar, que cualquier solución, algoritmo, máxima o buenas prácticas no constituyen un patrón. Aun cuando estén presentes todos los requisitos que establezcan los elementos integrantes de un patrón, no debe ser considerado como tal, hasta que se verifique que este es un **fenómeno recurrente**. El comportamiento recurrente debe estar presente al menos en tres sistemas existentes —**regla de tres—**, que una vez logrado no garantiza la calidad y fiabilidad total del patrón. Posteriormente será sometido a un profundo escrutinio, crítica y revisión por parte de la comunidad de patrones de software, que finalmente acreditará la validez del mismo [4].

La terminología patrón sugiere recurrencia, si algo no es recurrente no es posible que sea un patrón. La recurrencia, a pesar de ser necesaria, no es determinante para evaluar la importancia de un patrón, pues solamente lo caracteriza cuantitativamente. "Es necesario además demostrar que este sea apropiado para el uso, describiendo cómo el mismo es exitoso, y su utilidad mediante el por qué es beneficioso"[10].

La esencia principal para el surgimiento de los patrones, fue su poder de expresar conocimiento, con la documentación de buenas y probadas prácticas, mediante las diversas formas de representarlos. "Un libro de referencias compuesto por patrones documentando buenas prácticas pudiera proporcionar soluciones factibles para un determinado entorno. Un libro como este constituye una herramienta de invaluable utilidad práctica para la adquisición de dominios expertos" [19].

Cada patrón tiene que ser formulado en forma de regla que establezca la relación existente entre un contexto, un sistema de fuerzas que surjan a partir de dicho contexto y, una configuración o solución que permita que estas fuerzas sean despejadas entre sí en el contexto dado. Se recomienda, además, el empleo de ejemplos gráficos "(…)primeramente debe existir una imagen que muestre un prototipo de ejemplo del patrón" [1].

Diversos tipos de formatos formales han sido empleados en la documentación de los patrones. La descripción del patrón usada por Alexander en sus trabajos, es denominada **forma alexandrina**, también se encuentra la **forma GoF**, la **forma canónica** entre otras; a menudo sólo se diferencian unas de las otras por leves adaptaciones. No obstante, a pesar del empleo de diferentes formatos, generalmente todos convergen en determinados aspectos que constituyen los elementos indispensables para la correcta descripción de un patrón [4].

Existen, además, los formatos informales o mínimos, en los cuales la estructura reducida es apropiada para la documentación mínima del patrón. Son empleados en consultas en línea donde la viabilidad es clave y las formalidades no son tan necesarias; o en determinados medios de publicación cuyos requerimientos exijan contenidos escuetos. Entre estos formatos están las **formas micro**, **formas mini-inductivas** y las **formas mini-deductivas** [6].

Según Rising y Appleton, para todo tipo de formato formal que exprese la estructura de un patrón, determinados elementos —nombre, problema, contexto, fuerzas, solución, ejemplos, contexto resultante, patrones asociados y usos conocidos— deben ser reconocibles claramente en el momento de leerlos [19, 4]. Además, aunque no es estrictamente requerido, a menudo las descripciones de un patrón comienzan con un resumen, que proporciona un breve sumario o visión general del mismo al lector; una imagen más clara del patrón, rápida revelación de la información que corresponda a su utilidad o aplicabilidad en un problema dado. "Todo patrón debe identificar el objetivo al cual está dirigido y poner bien en claro qué debe ser asumido por el lector" [4].

Como objetivo fundamental, de acuerdo con una estructura correcta y bien escrita, cada patrón debe mostrar, en su totalidad, que es más grandioso que la suma de sus partes, y que debido al trabajo en grupo de sus elementos puede satisfacer diversas demandas.

AntiPatrones

Si los patrones representan las buenas prácticas, los antipatrones representan las lecciones aprendidas. "El estudio de antipatrones es una actividad investigativa de crucial importancia. La presencia de buenos patrones en sistemas exitosos no es suficiente; tiene que ser demostrado además que los mismos están ausentes en los sistemas fallidos. De igual manera,

es útil encontrar que ciertos antipatrones están presentes en sistemas fallidos, así como su ausencia en los que funcionan correctamente" [7]. Un antipatrón es, literalmente, una forma de describir la ocurrencia común de soluciones a problemas que degeneren irremediablemente en consecuencias negativas. Los antipatrones pueden ser el resultado de un ente administrativo o desarrollador, sin los suficientes conocimientos y experiencia en la solución de un problema particular, o habiendo aplicado un buen patrón en el contexto equivocado.

El concepto de antipatrón es una de las mayores atracciones de investigación dentro del campo de la Ingeniería del Software, centrada particularmente en el análisis de soluciones negativas. Dados los frecuentes defectos de aplicaciones de software y proyectos fallidos, las soluciones negativas son probablemente un objetivo de estudio mucho más rico. En estas investigaciones sobre antipatrones son categorizadas, etiquetadas y descritas soluciones negativas recurrentes; también son adjuntados determinados patrones que proporcionan alternativas constructivas que resuelven los orígenes del problema. Constituyen elementos de valor extraordinario y utilidad en circunstancias de recuperación, refactorización y reestructuración hacia buenas prácticas, a partir de situaciones adversas [6, 4].

En la figura 1 puede apreciarse que patrones y antipatrones son conceptos que se relacionan.

La esencia de un antipatrón se representa en dos soluciones, a diferencia de la pareja problema/solución encontrada en los patrones. La primera solución es problemática —ocurrencia de una solución que genera consecuencias muy negativas—. La segunda

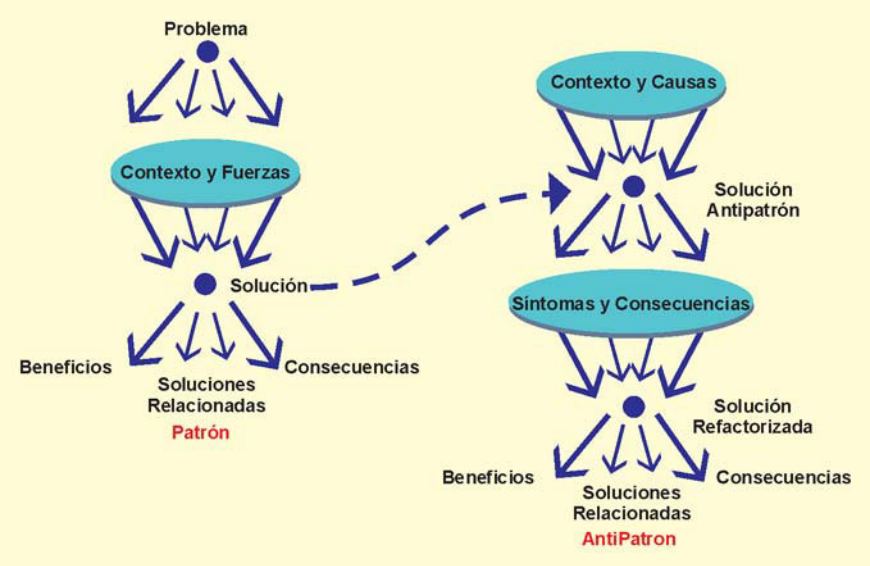

Figura 1 Relación de los conceptos patrón y antipatrón

solución, denominada solución refactorizada, que contrasta con la primera, consta comúnmente de la aplicación de un método con el cual el antipatrón puede resolverse teniendo en cuenta su ingeniería a favor de una forma más provechosa [6].

Como los patrones y antipatrones están relacionados, algunos patrones en ciertas circunstancias degeneran en antipatrones, "(…) un patrón muy popular, como es el caso de la programación estructurada, pudo ser un paradigma para su época, y decaer a favor de otro, una vez comprendidas las consecuencias desfavorables de su actual aplicación" [22]. La principal diferencia entre las soluciones de patrones y antipatrones radica en el contexto. Un antipatrón es un patrón aplicado en un contexto inadecuado. Cuando un patrón se convierte en antipatrón, es útil el conocimiento de dicho evento porque proporciona una señal de alarma, que notifica que la solución actual debe evolucionar hacia otra mejor. Este proceso de evolución, cambio o migración es denominado refactorización. En una refactorización, se cambia una solución por otra con una estructura mejorada que proporcione un incremento considerable de los beneficios [6].

La legibilidad es una de las cuestiones claves para la descripción de un patrón; en determinadas ocasiones es objeto de tediosas abstracciones donde los antipatrones llevan ventaja. En un antipatrón, el problema es representado sobre la base de errores comúnmente ocurridos, las soluciones erradas maximizan las catástrofes que subyacen bajo un problema medular. Esta maximización del problema constituye un paso primordial para el esclarecimiento de determinados propósitos. "Citando como ejemplo una prueba de software, el responsable de probar un determinado proceso, maximiza los errores a un nivel de colisión del sistema, con el objetivo de ganar la atención de los desarrolladores" [5].

La descripción de los antipatrones se fundamenta en una estructura retórica, diferente a la encontrada en los patrones. Estos comienzan con una apremiante y complicada solución, posteriormente ofrecen una

solución alternativa para refactorizar el problema. No se garantiza que esta solución sea única; pero es una vía efectiva para dilucidar las fuerzas que aseguran mejores beneficios. En cuestiones prácticas, los antipatrones son encontrados como formas poderosas y efectivas para la resolución de problemas recurrentes de los patrones.

Los antipatrones son nuevas formas de patrones, que por las diferencias de su estructura son descritos de una forma particular. En un patrón ordinario, la existencia, al menos, de tres usos conocidos de la solución es convencionalmente necesaria. Debido a que en los antipatrones existen dos soluciones en vez de una, esta regla de tres es diferente. La primera solución, la negativa, tiene que conformar las ocurrencias de dicha regla, justamente igual que en los patrones convencionales.

Desafortunadamente, es poco probable que cada una de estas tres ocurrencias de soluciones problemáticas sean resueltas exactamente de la misma manera. La segunda solución o refactorización está basada sobre la conocida solución negativa y, en caso de que sean múltiples, todas deben estar representadas correctamente en la formulación del antipatrón [6].

Al igual que en los patrones, existen diferentes formas formales e informales de representar a los antipatrones. En una representación formal, la plantilla completa de un antipatrón debe comprender, entre otros aspectos, las dos secciones centrales correspondientes a su descripción: la forma general del antipatrón y su correspondiente refactorización.

Un primer paso hacia la recuperación es admitir que se presenta un problema, los antipatrones ayudan a los afectados a clarificarlo apartándole de términos dramáticos. Pueden acceder a un vasto campo de aplicaciones, enfrentándose a los síntomas del problema, así como a las consecuencias de sus situaciones. “Muchas personas encuentran entretenido el uso de los antipatrones; el error es una cualidad humana; se ríen de sus propios errores y de los errores de otros, siempre y cuando no implique cualquier tipo de insulto a sus colegas” [6].

Puntualizando en la importancia y los beneficios que reporta para toda organización el reconocimiento de antipatrones dentro de sus sistemas, Thomas Mowbray y Raphael Malveau ofrecen los siguientes criterios: “Un correcto y actualizado conocimiento sobre AntiPatrones, es esencial para que el desarrollo de software sea exitoso” [16].

Son más los proyectos de software que fracasan que los que tienen éxito. Malos diseños, decisiones y proyectos prevalecen ante los buenos. El mundo real del software está colmado de antipatrones, coexistiendo irónicamente con soluciones altamente efectivas. Los malos diseños de software son, por lo general, el resultado de malas interpretaciones y errores clásicos. Los antipatrones explican por qué ocurren estos malos software, cómo refactorizarlos y cómo evitar la reiteración de los errores. Además, enseñan a identificar, arreglar errores a tiempo y prevén serias consecuencias.

La tecnología del software comercial está plagada de defectos, contradicciones y falsas promesas. Los antipatrones revelan esta realidad desde sus perspectivas internas, documentan la información necesaria sobre tecnologías pobres y mal fundamentadas para sobrevivir al emergente desarrollo industrializado de aplicaciones comerciales de software.

Muchos proyectos de software son caóticos, impredecibles y azarosos de conducir. Los antipatrones explican cómo los proyectos de software funcionan y cómo manejar correctamente sus evitables consecuencias; establecen una definición clara de las conductas negativas en el desarrollo de software y constituyen útiles descripciones de las buenas prácticas a las que toda organización debe migrar.

A pesar de las ventajas del uso de antipatrones en prácticas organizativas, deben ser cuidadosamente manejados, fundamentalmente en aquellos entornos que sean operativamente productivos. Muchas compañías y empresas son objeto de un sinnúmero de antipatrones, sin embargo, la ausencia de estos no garantiza que dichas organizaciones sean exitosas. De hecho, algunas lo son a pesar de repetidas violaciones de antipatrones. Ciertamente, la resolución de todo antipatrón puede arrojar una mejoría sustancial de las condiciones productivas, pero desgraciadamente no siempre es así. “Implementar una solución es factible solamente cuando el personal técnico tiene las suficientes aptitudes para afrontar el problema. Cuando los conocimientos técnicos no son suficientes para implementar plenamente una solución AntiPatrón, el remedio puede causar mayores y peores problemas que la afección original” [16].

Clasificación y organización de patrones

Para mejorar la comprensión de los patrones y utilizarlos eficientemente se han clasificado en cinco tipos, en dependencia del escenario en que son aplicados: pa-

trones de desarrollo, de análisis, organizacionales, de procesos y de dominios específicos.

Muchos de los enfoques iniciales de las investigaciones relacionadas con el tema han estado dirigidos, fundamentalmente, a los patrones de desarrollo. Debido a la abrumadora aceptación del texto comúnmente denominado GoF —*Gang of Four*—, han constituido la mayor fuente de consulta para la comunidad de software internacional. Según como los define el GoF, "(…) los patrones de desarrollo constituyen descripciones de las relaciones y comunicaciones entre objetos y clases, hechas a la medida para solucionar un problema de diseño general, enmarcado en un contexto particular dado" [11].

Por el gran volumen que representan, ha sido necesaria la subclasificación y organización de los patrones teniendo en cuenta los elementos que los caracterizan [4]. Son agrupados de acuerdo con las relaciones que tengan con un área tecnológica determinada. Dentro de cada grupo son desplegados de modo que puedan ser reconocidos, atendiendo a diferentes niveles de abstracción, para que los grupos de usuarios puedan encontrar los patrones que corresponden estrechamente con su área de interés; así como por diferentes puntos de vista que comprendan una misma solución. La combinación de los tres niveles de refinamiento en el eje vertical y los cuatro puntos de vista en el eje horizontal deriva una organización en forma de cuadrícula del grafo de patrones (Figura 2). Esta organización es conocida como Marco de Patrones.

Además de patrones de desarrollo son encontrados los patrones de análisis, definidos como "patrones que reflejan la estructura conceptual de un proceso, recurriendo a modelos de análisis reusables" [9]. En las etapas de análisis tratan de comprenderse los principales problemas involucrados en un proceso, con observación a través de los requerimientos que afloran en la modelación del mismo. Los patrones de análisis contribuyen a una mayor reusabilidad y calidad en el software en comparación con otras variedades de patrones, además simplifican la búsqueda de las estructuras óptimas de los sistemas, pues la base de la edificación de toda aplicación es el análisis. "La correcta modelación de un subsistema, correspondiente a un sistema complejo, puede ser abstraída y convertida en un patrón de análisis que pueda ser reutilizado por otras aplicaciones" [8, 4].

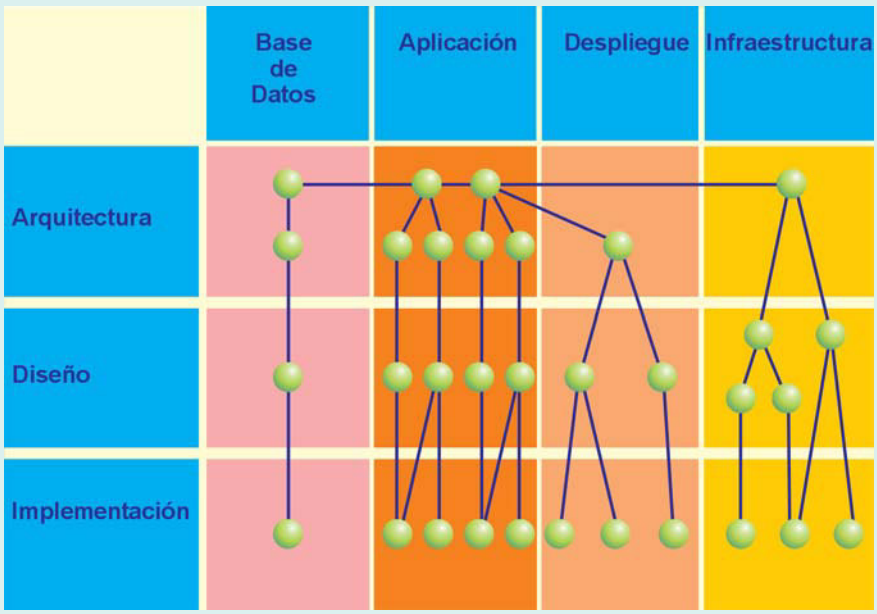

Figura 2 Marco de patrones

Los patrones organizacionales son aquellos que intervienen en los tópicos organizativos que se originan en los equipos de desarrollo de software, grupos y departamentos. Describen buenas técnicas de administración y la estructuración organizativa de las mismas. "La comprensión y modelación del contexto organizacional del entorno operativo de un sistema de software, constituye una de las primeras etapas de su análisis preliminar. Es conveniente la elaboración de estos modelos tomando como herramientas, el despliegue y utilización de patrones organizacionales, que describan a menudo usadas, óptimas y bien probadas estructuras organizativas"[15]. Actualmente, los patrones de desarrollo parecen ser los más populares, pero los organizacionales ganan en impulso.

La descripción de técnicas generales, acciones y tareas para la conducción y desarrollo del proceso inherente a un proyecto de software, constituye esencialmente la función de los patrones de procesos. Un importante rasgo que caracteriza a los patrones de procesos es que describen qué se debe hacer; pero no detallan cómo realizar una determinada acción. "Cuando son aplicados integralmente de una forma organizada,

los patrones de procesos pueden ser útilmente utilizados para la construcción de los procesos relacionados al desarrollo de aplicaciones de software en organizaciones dedicadas a la actividad. Dado que los mismos no especifican profundamente cómo realizar una determinada tarea, los patrones de procesos constituyen los reusables bloques constructivos con los que toda organización podrá edificar el proceso de desarrollo de software hecho a la medida de sus necesidades" [3]. Los patrones organizacionales se encuentran muy relacionados con los de procesos.

Dentro de la clasificación de patrones de dominios específicos han sido enmarcados aquellos que, como los define su nombre, están presentes dentro de dominios especializados, que no intervienen directamente en el desarrollo de software; pero por su importancia ejercen una acción indirecta en la calidad de los mismos. Entre estos tipos se encuentran los patrones referentes a la pedagogía, educación y entrenamiento de tecnologías del software, arquitectura e infraestructura de redes, entre otros [4].

Beneficios y obligación en el uso de patrones

Los patrones son un punto de contacto, en el que se consolidan y complementan la teoría y la práctica, muestran útil, aplicable y explotada la estructura que conjuntamente describen. "Es conformado un continuo ciclo en el que especialistas dedicados a la investigación, documentan patrones que son puestos bajo la consideración y aprobación de expertos y posteriormente empleados por practicantes y desarrolladores en el mundo real, mejorando considerablemente la calidad de los productos finales" [4].

Existen beneficios inherentes a la adopción de los patrones en el desarrollo de aplicaciones de software [21].

Los patrones posibilitan la comunicación entre especialistas a un elevado nivel de abstracción; introducen un nuevo vocabulario, donde cada nombre representa una palabra, con todos los detalles correspondientes a su descripción. Analistas, arquitectos y diseñadores podrían discutir temas referentes a un proyecto con el empleo de este lenguaje, sin necesidad de caer en los detalles específicos. Esto ayuda a la toma de decisiones, con mayor facilidad y eficiencia en cuanto a diversos temas del proceso de desarrollo.

Los patrones facilitan la comunicación del desarrollo de software entre los integrantes de un proyecto. El lenguaje de patrones permite a los grupos de trabajo compartir los diseños de los subsistemas y las interfases existentes. Grupos diferentes podrán alcanzar una mejor comprensión del dominio del sistema a desarrollar. Además, facilitan la comunicación del diseño de la arquitectura entre desarrolladores, dirigentes, estrategas de mercado y, en general, entre todos los roles de una organización. Un vocabulario común permite a las personas con roles de menor nivel técnico comprender mejor la arquitectura del sistema sin necesidad de involucrarse en los detalles particulares.

En la documentación no será necesario detallar aspectos del diseño porque el nombre de los patrones utilizados dará indirectamente esta información. Por lo tanto, la documentación será más corta y llevará menos tiempo escribirla, revisarla y darle mantenimiento.

Los patrones permiten la reutilización de la arquitectura y el diseño de un sistema, y deben ser independientes de la plataforma. Entonces, una arquitectura y un diseño que utilicen patrones serán también independientes de la plataforma y permanecerán estables hasta que se realicen cambios en la plataforma de bajo nivel. Solamente será necesario portar la implementación de la arquitectura y el diseño.

Los patrones brindan conocimientos expertos a los menos instruidos. Especialistas con menor experiencia tendrán acceso a soluciones estables y bien diseñadas por expertos, lo que constituye gran ayuda para nuevos diseñadores de software y para los desarrolladores experimentados que incursionan en dominios novedosos.

A pesar de las virtudes que las diferentes definiciones sobre los patrones destacan, su uso requiere de un profundo estudio, disciplina y responsabilidad colectiva dentro de cualquier entorno de desarrollo de software que los utilice. "Los patrones no crean diseñadores de novatos o estudiantes de forma mágica, no es una bala de plata que convierta en gurú a cualquiera en una organización" [20].

Las siguientes consideraciones constituyen obligaciones y precauciones de probada validez, vinculadas a la utilización práctica de los patrones [13].

Uno de los beneficios del empleo de patrones es la facilidad que brindan en la comunicación del diseño. Para obtenerlo, los participantes de una organización deberán encontrarse familiarizados con los patrones empleados. Si alguien no los conoce, irremediablemente

fracasará la comunicación. Por lo tanto, implica un constante incremento del adiestramiento en los participantes del proyecto. Nuevos patrones serán escritos y patrones existentes podrán cambiar, de ese modo, debe realizarse un seguimiento en la educación sobre estas modificaciones.

Los especialistas pueden confundir un patrón con otro o simplemente confundir los contextos donde se desenvuelven. Esto puede fracasar la comunicación, pues los integrantes de un equipo discutirán asuntos diferentes sin darse cuenta de ello, y lo que es peor, puede ser tomada una decisión errónea.

Algunos especialistas querrán utilizar un patrón por el mero hecho de hacerlo, sin necesidad alguna; otros pudieran utilizar tantos patrones como les sea posible, sin considerar si pueden ser aplicados al problema en análisis. Esto conllevará a un diseño de la estructura del proyecto más complejo que si los patrones no hubieran sido utilizados.

Conclusiones

En el estado actual de la industria del software es un hecho la imposibilidad de producir, sin la ayuda de la Ingeniería del Software, con los índices de calidad y eficiencia adecuados. Sin el conocimiento y empleo de modelos, metodologías, técnicas y herramientas, inevitablemente son encontrados problemas con la productividad de los trabajadores, los tiempos de entrega de los productos y de la documentación, la calidad de las pruebas y la falta de comunicación efectiva entre los involucrados en una organización, grupo o proyecto.

El uso de patrones y antipatrones por especialistas de la informática, constituye una de las mayores fuentes de aprendizaje, perfeccionamiento individual y colectivo, comunicación y consolidación del proceso de desarrollo de software. Aunque no son las únicas técnicas disponibles para perfeccionar el software en la actualidad, gracias a su poder de expresión, dinamismo y adaptación a toda organización, las convierten en uno de los mejores métodos de hacer Ingeniería de Software para el presente y el futuro.

Referencias y Bibliografía

[1] Alexander, C. *The Timeless Way of Building*. USA: Oxford University Press, 1979.

[2] Alexander, C.; Ishikawa, S.; et al. *A Pattern Language: Towns, Buildings, Construction*. USA: Oxford University Press, 1977.

[3] Ambler, S. W. *An Introduction to Process Patterns*. Disponible en: http://www.ambysoft.com/processPatterns.pdf. (1998) (Consultado: febrero, 2004).

[4] Appleton, B. *Patterns and Software: Essential Concepts and Terminology*. Disponible en: http://www.bradapp.net/. (2000) (Consultado: julio, 2003).

[5] Bezier, B. *Foundations of Testing Computer Software*. 14th International Conference and Exposition on Testing Computer Software. (1997).

[6] Brown, W. H.; Malveau, R. C.; et al. *AntiPatterns. Refactoring Software, Architectures, and Projects in Crisis*. New York: John Wiley & Sons, Inc., 1998.

[7] Coplien, J. O. *A Generative Development-Process Pattern Language*. "The Pattern Handbook. Techniques, Strategies, and Application". UK: Cambridge University Press, 1996.

[8] Fernández, E. B. *Building Systems Using Analysis Patterns*. Disponible en: http://www.cs.wpi.edu/~cs562/s99/resources/ISAW-3/fernandez.pdf. (1999) (Consultado: enero, 2004).

[9] Fowler, M. *Analysis Patterns: Reusable Object Models*. USA: Addison-Wesley, 1997.

[10] Gabriel, R. *Patterns of Software: Tales from the Software Community*. (1998)

[11] Gamma, E.; Helm, R.; et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. USA: Addison - Wesley, 1995.

[12] Gray, J. *Software Engineering Tools*. 33rd International Conference on System Sciences, 2000.

[13] Helm, R. *Patterns in Practice*. Conference on Object Oriented Programming Systems Languages and Applications. OOPSLA '95. ACM Press,1995, págs. 337-341.

[14] IEEE. *Standards Collection Software Engineering. IEEE Standard 610.12-1990*. (1993).

[15] Kolp, M., Castro, J., et al. *Organizational Patterns for Early Requirements Analysis*. Disponible en: http://www.cs.toronto.edu/km/tropos/re01-f.pdf. (2001) (Consultado: enero, 2004).

[16] Mowbray, T. J. and Malveau, R. C. *CORBA AntiPaterns and Design Patterns*. New York: John Wiley & Sons, Inc., 1999.

[17] Paulk, M. C., Weber, C. V., et al. *CMM: The Capability Maturity Model*. Software Engineering Institute. Carnegie Mellon University, 1995.

[18] Pressman, R. S. *Software Engineering: A Practitioner's Approach. 5 ed.* USA: McGraw-Hill, 2001.

[19] Rising, L. *Design Patterns: Elements of Reusable Architecture*, 1998.

[20] Rising, L. *The Patterns Handbook. Techniques, Strategies, and Applications*. UK: Cambridge University Press,1998.

[21] Schmidt, D. C. "Using Design Patterns to Develop Reusable Object-Oriented Communication Software". *Communications of the ACM*, vol. 38, no. 10 (1995): 65-74.

[22] Webster, B. F. *Everything You Know is Wrong*. Object World West '99. SOFTBANK-COMDEX, 1999.

Nota editorial: en este artículo, hemos decidido hacer una excepción en relación con las normas para citas, notas o referencias bibliográficas y la bibliografía de nuestra publicación. Por su particularidad y complejidad, hemos respetado la forma en que las ha utilizado el autor.